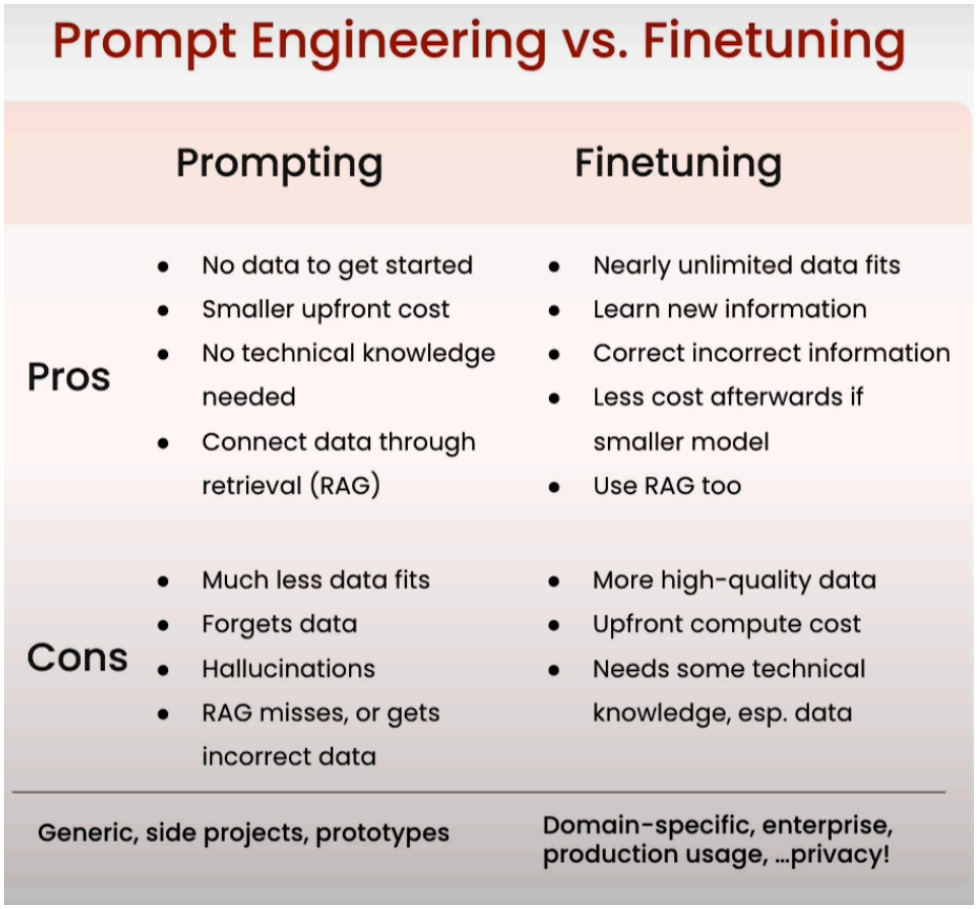
Fine tuning an LLM:

Taking a generic LLM and specializing it in the task at hand

Prompting Vs Finetuning:



Fine Tuning our own LLM:

privacy , lower cost per request, reduce latency, control uptime, moderation (If we want the model to provide some custom response or another customized response)

Where finetuning fits in?

Pre-training:
A random model that has no knowledge of the world. It cannot form English words. It does the next token prediction. But, it is taking in large amounts of data scraped from the web and often unlabelled. This is generally called self-supervised learning as it does not have labels to check with. After training it learns language and knowledge.

Pretraining

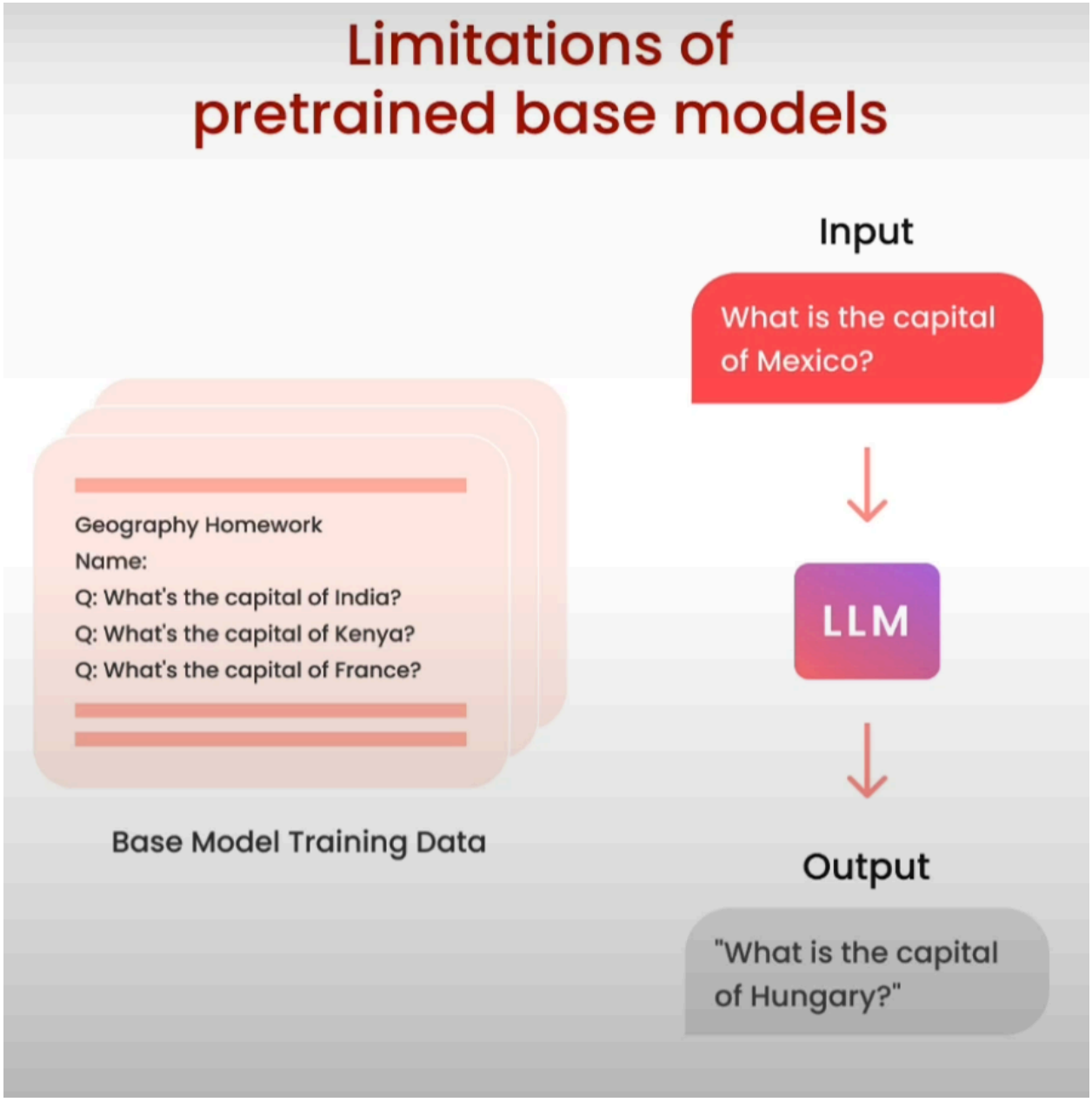
- Model at the start:
  - Zero knowledge about the world
  - Can't form English words
- Next token prediction
- Giant corpus of text data
- Often scraped from the internet: "unlabeled"
- Self-supervised learning

- After Training
  - Learns language
  - Learns knowledge

What is data scraped from the internet??
There is an open-source pretraining data that has data from 22 sources of data. This is time-consuming and needs a lot of compute as the model has to go through a lot of data.

What is “data scraped from the internet”?

- Often not publicized how to pretrain
- Open-source pretraining data: "The Pile"
- Expensive & time-consuming to train

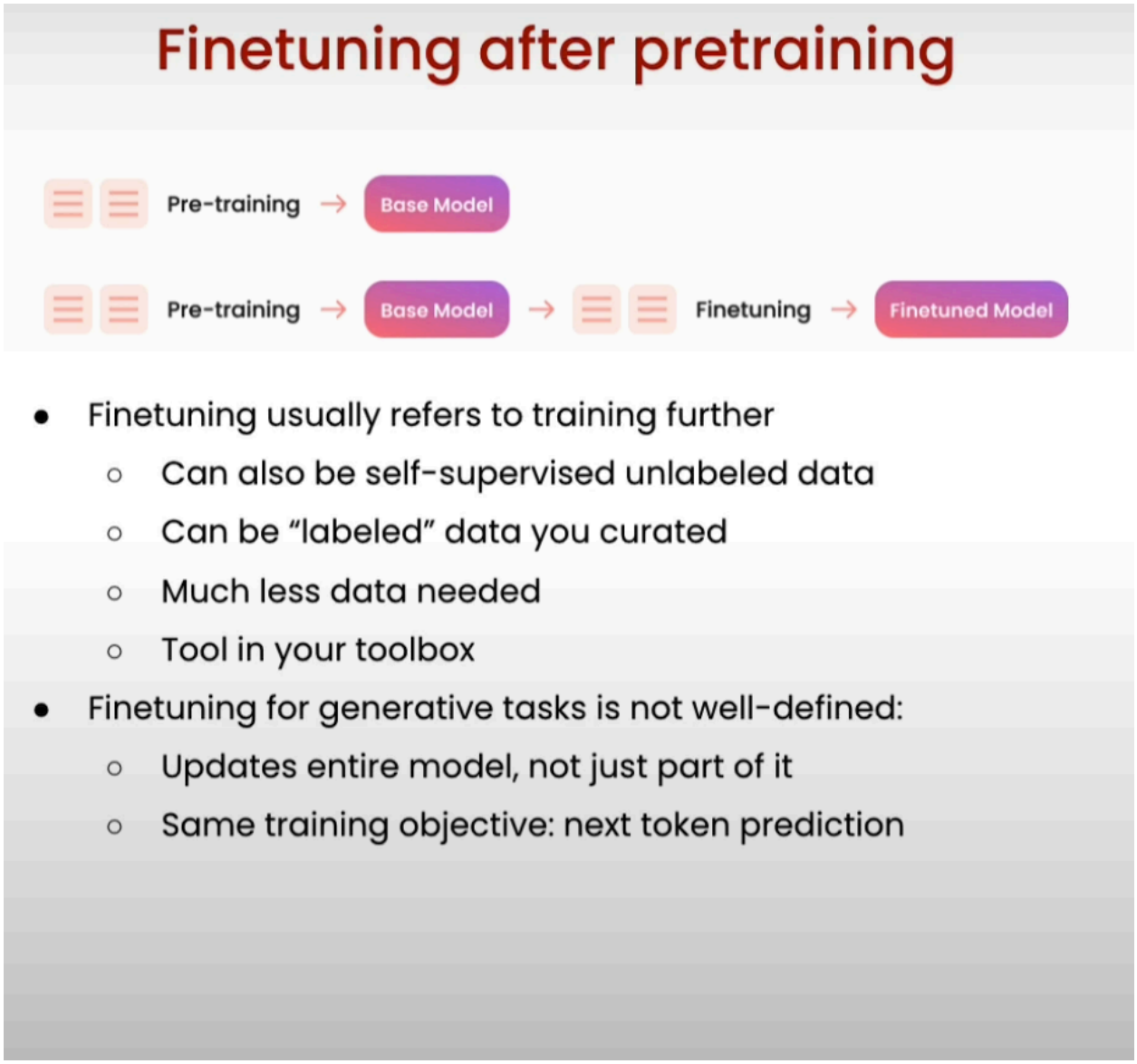


The base model just performs the task like above as it has seen in various instances in internet.

Finetuning after pretraining:

Later, we can use another scraped (self-supervised unlabeled data) or curated data (labeled data). Much less data is needed as we are already using a pretrained model.

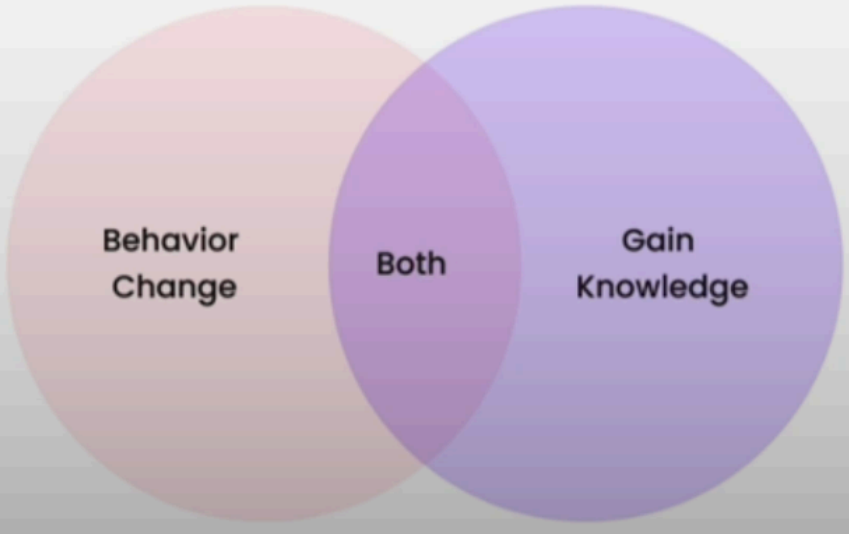
In finetuning, we often update the weights of the entire model and not just parts of it.



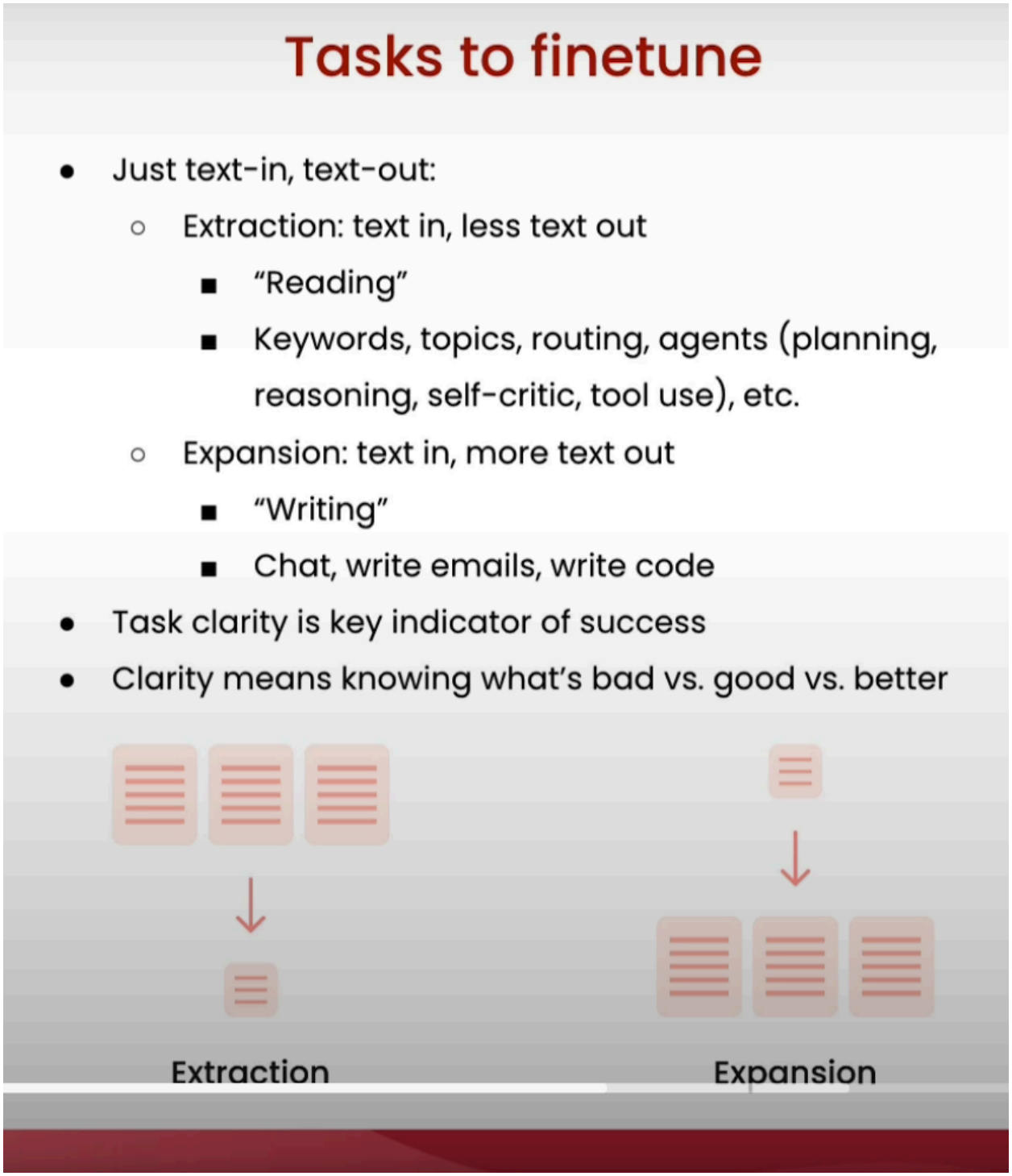
There are more advanced ways for us to decide how much of the model can we change.

What is finetuning doing for you?

- Behavior change
  - Learning to respond more consistently
  - Learning to focus, e.g. moderation
  - Teasing out capability, e.g. better at conversation
- Gain knowledge
  - Increasing knowledge of new specific concepts
  - Correcting old incorrect information
- Both



Tasks to finetune:



Extraction or Expansion?

What is a good o/p vs better or bad o/p

First time finetuning:

Lab Notebook:

1) We have scraped data
2) We have a dataset .json file with questions and answers pertaining to a specific company. We will be using this dataset to finetune.
3) We take questions and answers/ instructions and responses/ inputs and outputs and concatenate them and then serve the model.

   Sometimes we find that the contenation can do the trick. Sometimes it is not enough. Sometimes the model needs the data to have more structure in it to better understand

the data. So, we can further format the entire text to include sub-headings like Question: and Answer:.

```
prompt_template_qa = """### Question:
{question}

### Answer:
{answer}"""
```

```
question = examples["question"][0]
answer = examples["answer"][0]

text_with_prompt_template = prompt_template_qa.format(question=question, answ
text_with_prompt_template
```

```
"### Question:\nWhat are the different types of documents available in the re
pository (e.g., installation guide, API documentation, developer's guide)?\n
\n### Answer:\nLamini has documentation on Getting Started, Authentication, Q
uestion Answer Model, Python Library, Batching, Error Handling, Advanced topi
cs, and class documentation on LLM Engine available at https://lamini-ai.gith
ub.io/."
```

But, we are currently sticking to keep the questions and answers separate as it makes things better for us during evaluation.

```
prompt_template_q = """### Question:
{question}

### Answer:"""
```

We then apply this format to the entire dataset and bring the entire question and answers to be in this format.

Instruction finetuning:

A variant of GPT that made it into ChatGPT.

Instruction-following datasets

Some existing data is ready as-is, online:

- FAQs
- Customer support conversations
- Slack messages

LLM Data Generation

README

To authenticate, retrieve the API key from the Settings page.

Non-Q&A data can also be converted to Q&A

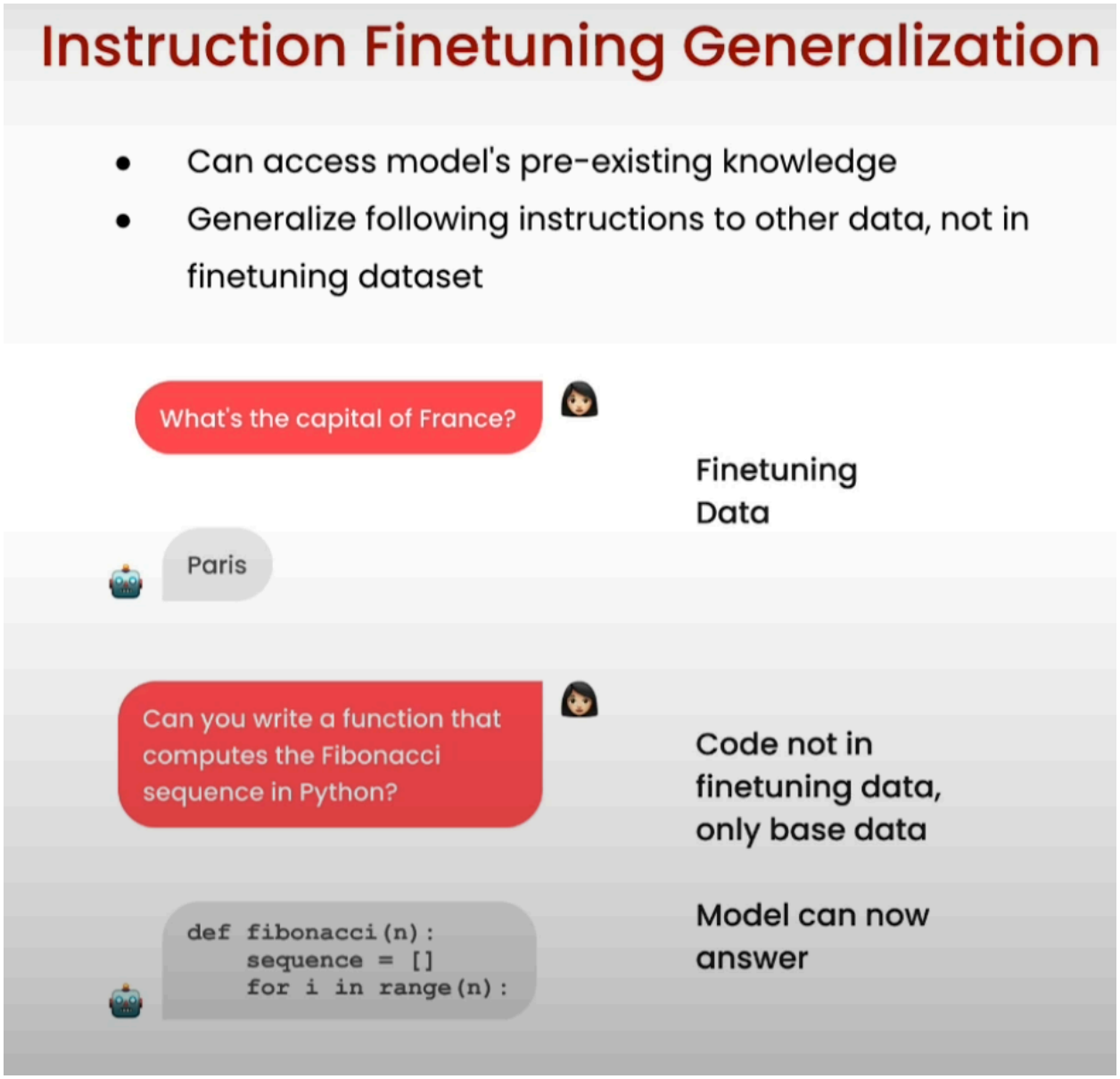
- Using a prompt template
- Using another LLM

LLM Generation Pipeline

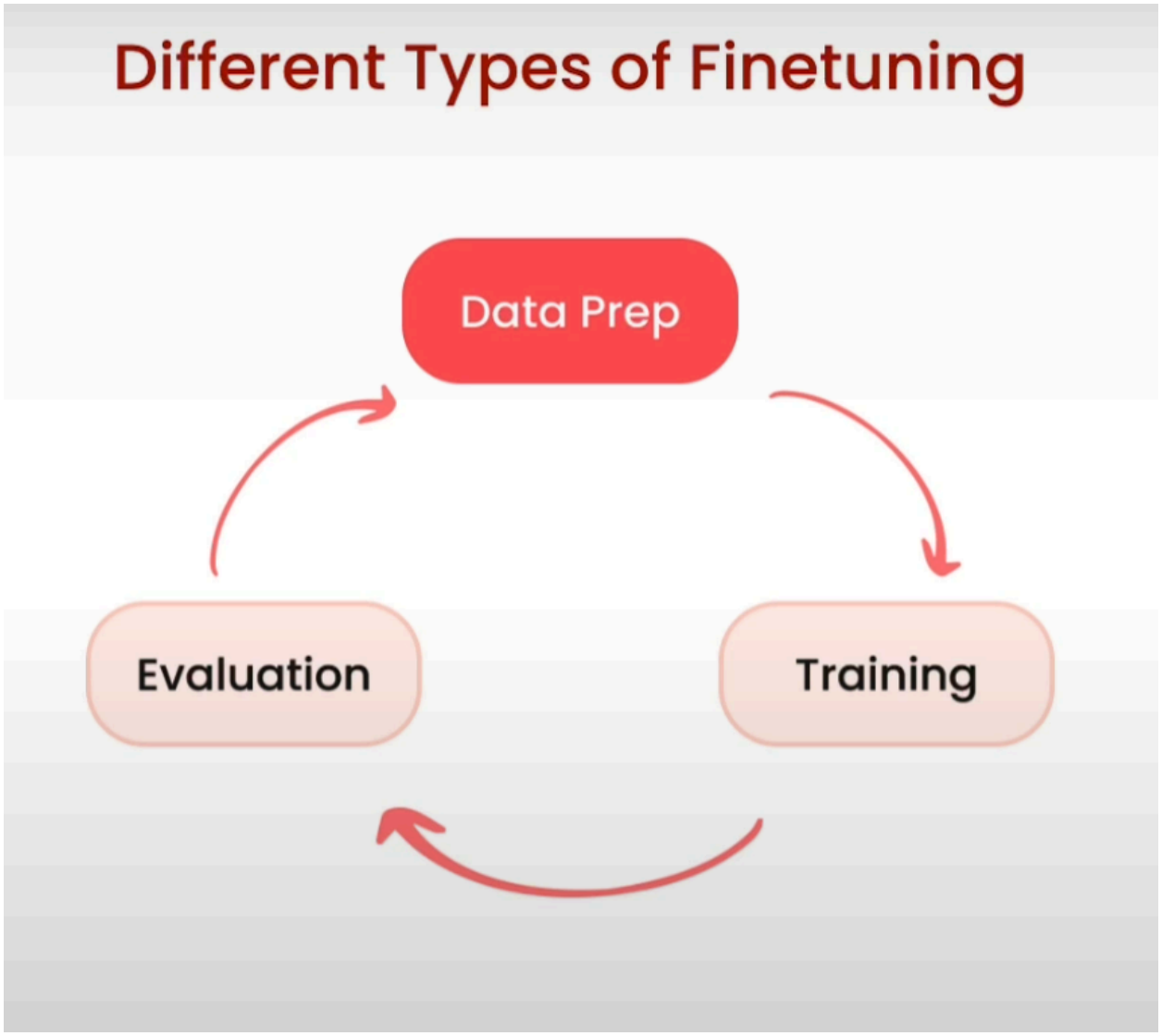
- ChatGPT ("Alpaca")
- Open-source models

How do you authenticate your request?

You must retrieve the API key from the Settings page.



The model can answer based on the data that we finetuned the model on. But, the model can also answer the questions about stuff other than what it is in the finetuning data, based on the pretraining that the model has been on.

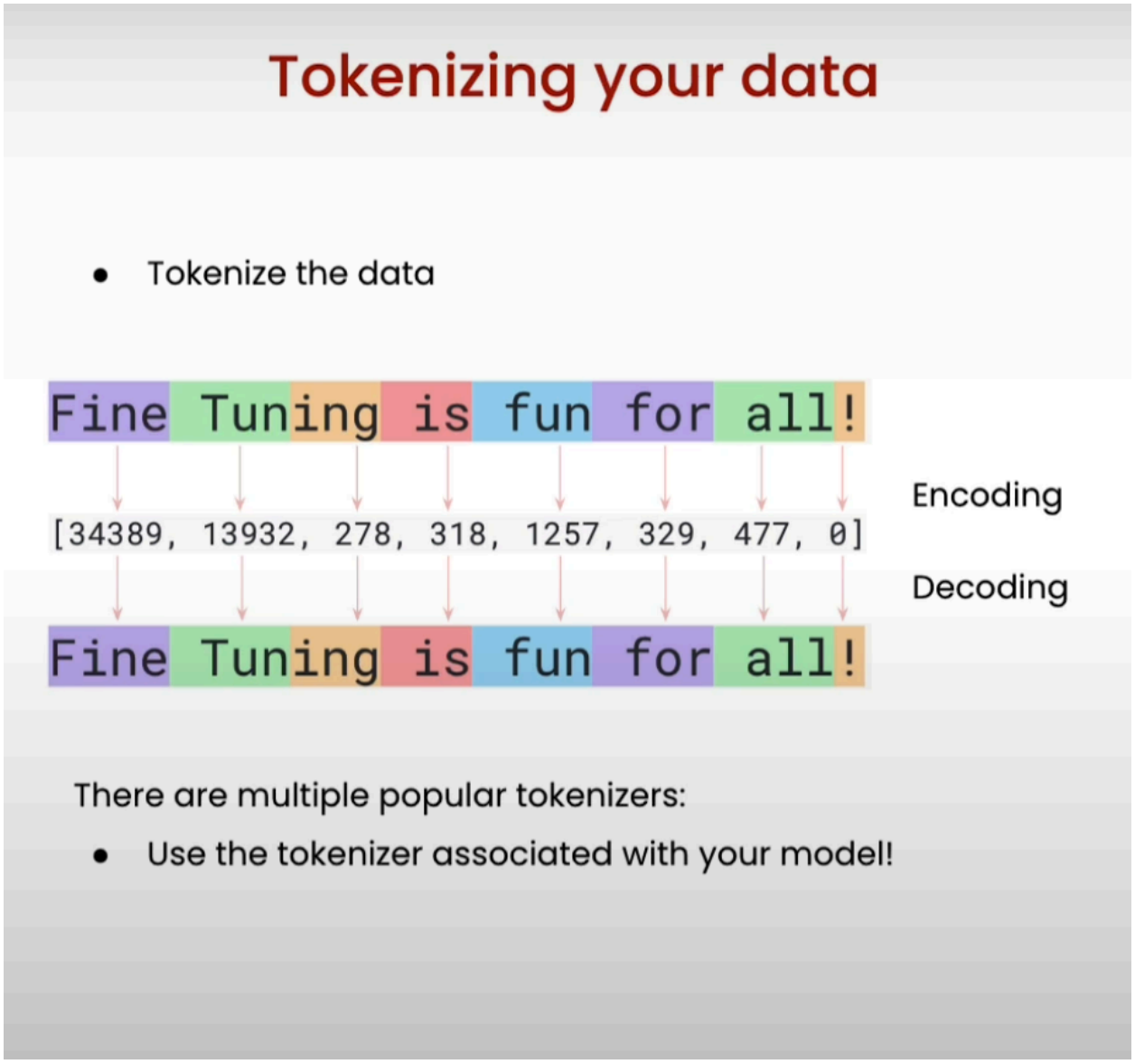


This Data Prep is what would be different based on what are we actually doing, are we dealing with questions/answers or instructions/responses etc…

Data Preparation:

Real data is always preferred to use than generated data as generated data often has an intrinsic pattern in it and the model gets it faster. However, that might not be the case in the case of real data.


Steps to prepare your data
1
Collect instruction-response pairs
2
Concatenate pairs
(add prompt template, if applicable)
3
Tokenize: Pad, Truncate
4
Split into train/test



It is important that when we are using a pretrained model, we are still using the right tokenizer that the model has been trained upon. This is important because, if we use a different tokenizer than the one used to train the model, the model will be confused about what are the actual tokens and their corresponding numbers.

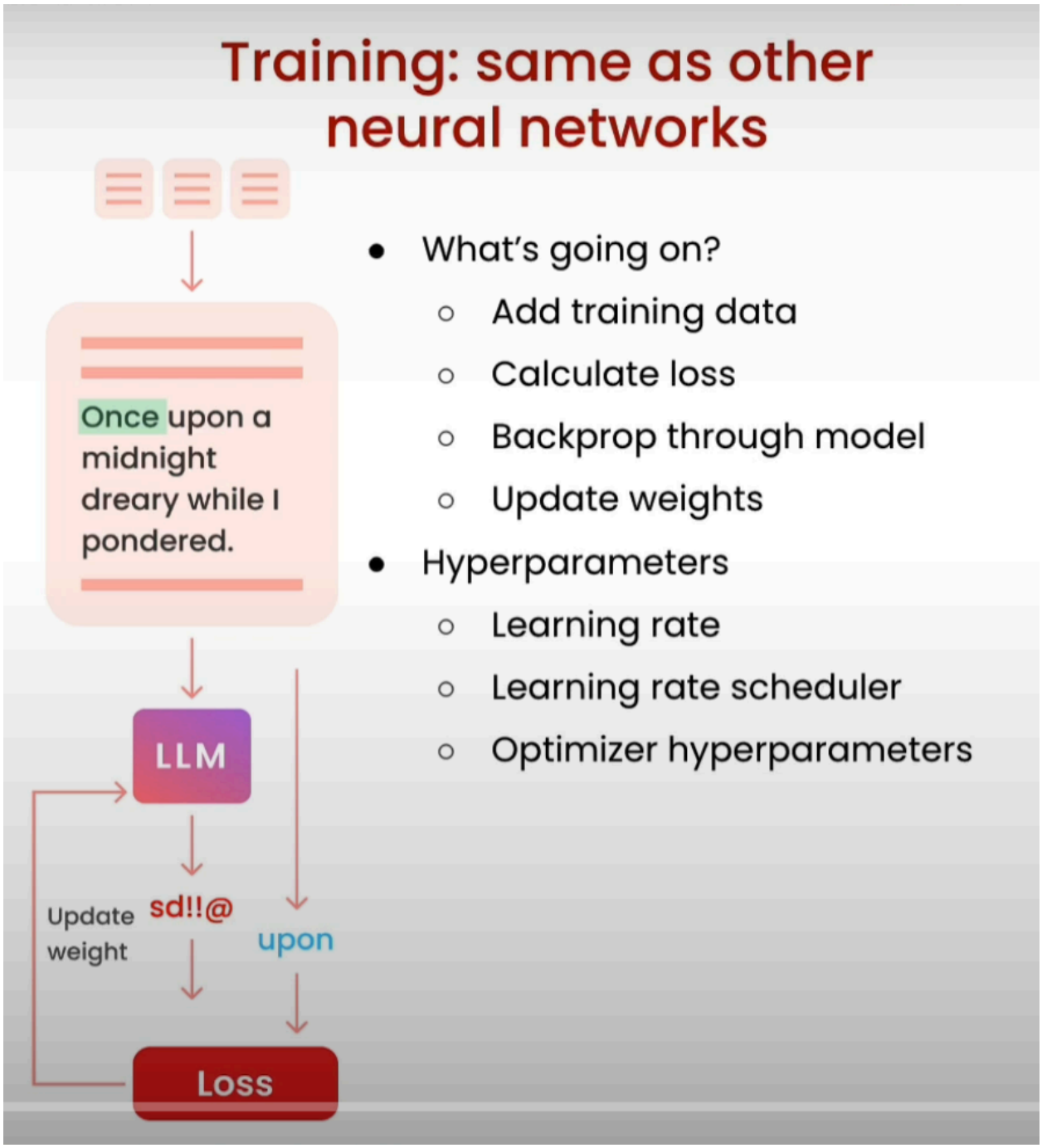
AutoTokenizer from Hugging face automatically uses the right tokenizer according to a model that we select.

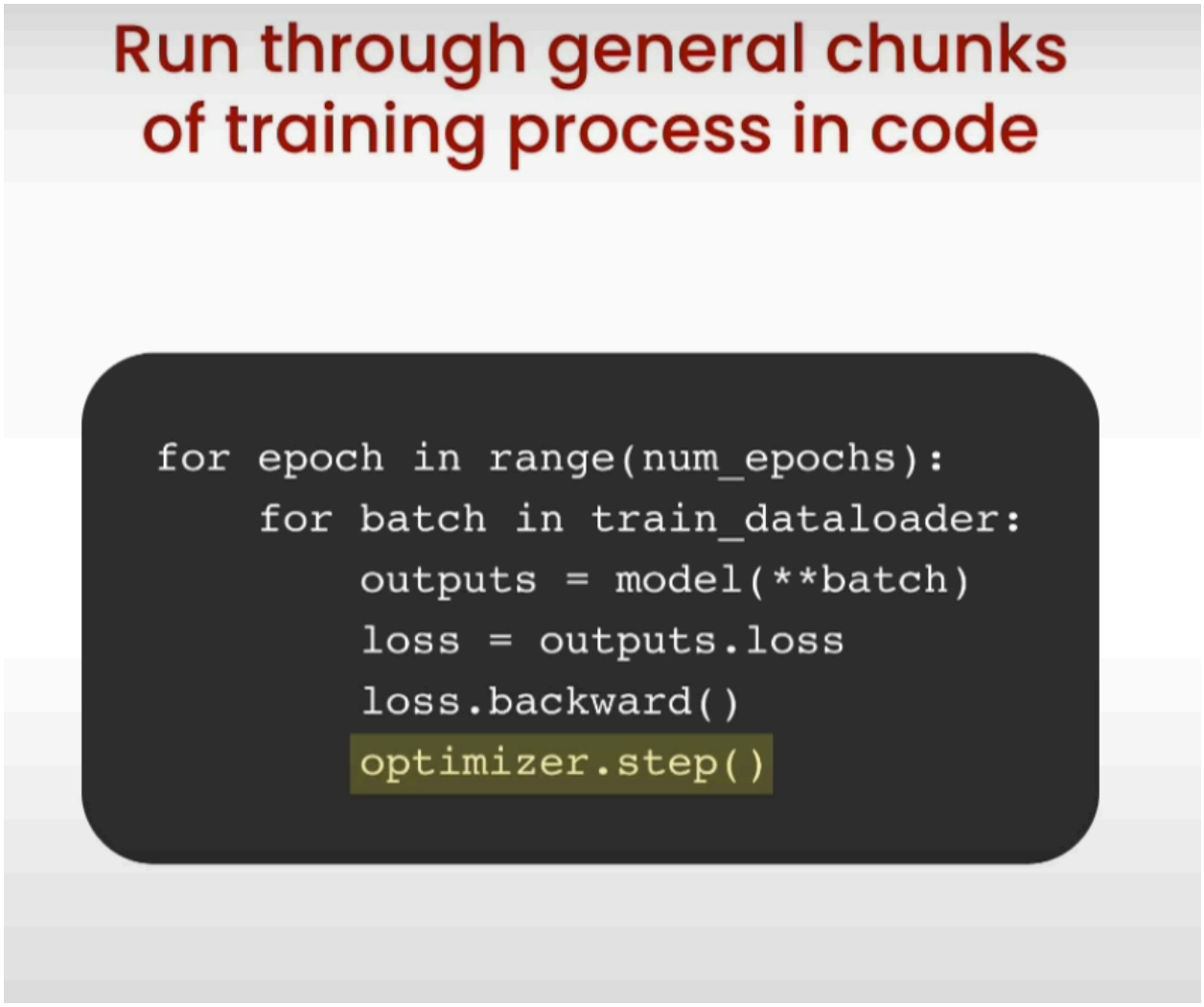
1) Tokenize
2) Padding and truncation
   We need all the inputs to the transformer typically to have the same length as this makes things easier to code. Hence, we pad shorter sentences and truncate longer sentences to a required length. However, truncating a sentence by force might lose some important

information as in most of the cases, the right side has more important information. Hence, we also truncate from the left side sometimes and this is task-specific.

3) Now, after tokenizing the entire dataset, we split the dataset into train and test datasets.

Training Process:





Function to carry out inference:

1) We tokenize the input text
2) Padding and truncating
3) Making sure the tokens are on the right device, cpu or gpu

Explore moderation using small model:
Encouraging the model not to get too off track.